WE CLAIM:

1. A method of executing a sequence of variable length instructions stored within a plurality of discrete memory address regions within a memory of a data processing apparatus, said method comprising the steps of:

(i) detecting an attempt to execute a variable length instruction spanning two discrete memory address regions, said two discrete memory address regions being a current memory address region and a following memory address region;

(ii) concatenating instruction data from an end portion of said current memory address region and a start portion of said following memory address region into a fix-up memory address region of said memory to form concatenated instruction data containing said variable length instruction;

(iii) diverting program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region; and

(iv) restoring program execution flow to execute instructions following said variable length instruction from within said following memory address region.

2. A method as claimed in claim 1, wherein said steps of detecting is performed under hardware control.

3. A method as claimed in claim 1, wherein said steps of concatenating, diverting and restoring are performed under software control.

4. A method as claimed in claim 1, wherein variable length instructions are fetched from said memory to an instruction buffer before being executed.

5. A method as claimed in claim 4, wherein said step of detecting occurs as said variable length instruction is read from said instruction buffer.

6. A method as claimed in claim 4, wherein fetching of variable length instructions to said instruction buffer is performed by fetching instruction data from sequential memory addresses under hardware control.

7. A method as claimed in claim 6, wherein when a fetch is attempted from a memory address beyond an end point of said current memory address region to a buffer memory location, said buffer memory location is marked as not containing valid instruction data.

8. A method as claimed in claim 7, wherein said step of detecting comprises detecting an attempt to execute an instruction at least partially stored in a buffer memory location marked as not containing valid instruction data.

9. A method as claimed in claim 1, wherein a program counter value specifies a location of a variable length instruction to be executed.

10. A method as claimed in claim 9, wherein said steps of diverting and restoring act by modifying said program counter value.

11. A method as claimed in claim 1, comprising setting a single step flag, said single step flag serving to limit hardware execution of variable length instructions to a single variable length instruction.

12. A method as claimed in claim 11, wherein said single step flag serves to limit hardware execution of variable length instructions to a single variable length instruction from said concatenated instruction data to a single variable length instruction before control is returned to software to perform said step of restoring.

13. A method as claimed in claim 11, wherein upon hardware execution of said single variable length instruction, said single step flag is cleared under hardware control.

14. A method as claimed in claim 11, wherein said single step flag is stored within a coprocessor register.

15. A method as claimed in claim 1, comprising calculating a start address within said following region of memory of a following variable length instruction following said current variable length instruction.

16. A method as claimed in claim 15, wherein said step of calculating uses as inputs a start address of said following memory region and a program counter value pointing to said following variable length instruction within said fix-up memory region following execution of said current variable length instruction.

17. A method as claimed in claim 16, comprising storing said start address of said following memory region before said step of diverting.

18. A method as claimed in claim 1, wherein said variable length instructions are Javacard bytecode instructions executed as native instructions by said data processing apparatus.

19. A method as claimed in claim 18, wherein said data processing apparatus also supports execution of instructions of a further instruction set, said steps of concatenating, diverting and restoring being performed under control of instructions of said further instruction set.

20. A method as claimed in claim 19, wherein said steps of diverting and restoring are performed using state switching branch instructions that serve to switch to execution of Java bytecodes starting from a specified memory address location.

21. A method as claimed in claim 1, wherein a program to be executed is stored within fragmented memory regions within said memory.

22. Apparatus for executing a sequence of variable length instructions stored within a plurality of discrete memory address regions within a memory, said apparatus comprising:

(i) a detector operable to detect an attempt to execute a variable length instruction spanning two discrete memory address regions, said two discrete memory address regions being a current memory address region and a following memory address region;

(ii) combining logic operable to concatenate instruction data from an end portion of said current memory address region and a start portion of said following memory address region into a fix-up memory address region of said memory to form concatenated instruction data containing said variable length instruction;

(iii) diverting logic operable to divert program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region; and

(iv) restoring logic operable to restore program execution flow to execute instructions following said variable length instruction from within said following memory address region.

23. Apparatus as claimed in claim 22, wherein said detector is non-programmable hardware.

24. Apparatus as claimed in claim 22, wherein said concatenating logic, said diverting logic and said restoring logic comprise programmable hardware operating under software control.

25. Apparatus as claimed in claim 22, wherein variable length instructions are fetched from said memory to an instruction buffer before being executed.

26. Apparatus as claimed in claim 25, wherein said detector acts as said variable length instruction is read from said instruction buffer.

27. Apparatus as claimed in claims 25, wherein fetching of variable length instructions to said instruction buffer is performed by fetching instruction data from sequential memory addresses under hardware control.

28. Apparatus as claimed in claim 27, wherein when a fetch is attempted from a memory address beyond an end point of said current memory address region to a buffer memory location, said buffer memory location is marked as not containing valid instruction data.

29. Apparatus as claimed in claim 28, wherein said detector is operable to detect an attempt to execute an instruction at least partially stored in a buffer memory location marked as not containing valid instruction data.

30. Apparatus as claimed in claim 22, wherein a program counter value specifies a location of a variable length instruction to be executed.

31. Apparatus as claimed in claim 30, wherein said diverting logic and said restoring logic act by modifying said program counter value.

32. Apparatus as claimed in claim 22, comprising setting a single step flag, said single step flag serving to limit hardware execution of variable length instructions to a single variable length instruction.

33. Apparatus as claimed in claim 32, wherein said single step flag serves to limit hardware execution of variable length instructions to a single variable length instruction from said concatenated instruction data to a single variable length instruction before control is returned to software to perform said step of restoring.

34. Apparatus as claimed in claim 32, wherein upon hardware execution of said single variable length instruction, said single step flag is cleared under hardware control.

35. Apparatus as claimed in claim 32, wherein said single step flag is stored within a coprocessor register.

36. Apparatus as claimed in claim 22, comprising calculating logic operable to calculate a start address within said following region of memory of a following variable length instruction following said current variable length instruction.

37. Apparatus as claimed in claim 36, wherein said calculating logic uses as inputs a start address of said following memory region and a program counter value pointing to said following variable length instruction within said fix-up memory region following execution of said current variable length instruction.

38. Apparatus as claimed in claim 37, comprising storing said start address of said following memory region before said diversion.

39. Apparatus as claimed in claim 22, wherein said variable length instructions are Javacard bytecode instructions executed as native instructions by said data processing apparatus.

40. Apparatus as claimed in claim 39, wherein said data processing apparatus also supports execution of instructions of a further instruction set, said steps of concatenating, diverting and restoring being performed under control of instructions of said further instruction set.

41. Apparatus as claimed in claim 40, wherein said diverting logic and said restoring logic use mode switching branch instructions that serve to switch to execution of Java bytecodes starting from a specified memory address location.

42. Apparatus as claimed in claim 22, wherein a program to be executed is stored within fragmented memory regions within said memory.

43. A computer program product for controlling a data processing apparatus operable to executing a sequence of variable length instructions stored within a plurality of discrete memory address regions within a memory of said data processing apparatus, said computer program product comprising:

code operable after an attempt to execute a variable length instruction spanning two discrete memory address regions, said two discrete memory address regions being a current memory address region and a following memory address region, said code including:

(i) concatenating code operable to concatenate instruction data from an end portion of said current memory address region and a start portion of said following memory address region into a fix-up memory address region of said memory to form concatenated instruction data containing said variable length instruction;

(ii) diverting code operable to divert program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region; and

(iii) restoring code operable to restore program execution flow to execute instructions following said variable length instruction from within said following memory address region.

44. A computer program product as claimed in claim 43, wherein detection of said attempt to execute a variable length instruction spanning two discrete memory address regions is performed under hardware control.

45. A computer program product as claimed in claim 43, wherein variable length instructions are fetched from said memory to an instruction buffer before being executed.

46. A computer program product as claimed in claim 45, wherein said detection occurs as said variable length instruction is read from said instruction buffer.

47. A computer program product as claimed in claim 45, wherein fetching of variable length instructions to said instruction buffer is performed by fetching instruction data from sequential memory addresses under hardware control.

48. A computer program product as claimed in claim 47, wherein when a fetch is attempted from a memory address beyond an end point of said current memory address region to a buffer memory location, said buffer memory location is marked as not containing valid instruction data.

49. A computer program product as claimed in claim 48, wherein said detection comprises detecting an attempt to execute an instruction at least partially stored in a buffer memory location marked as not containing valid instruction data.

50. A computer program product as claimed in claim 43, wherein a program counter value specifies a location of a variable length instruction to be executed.

51. A computer program product as claimed in claim 50, wherein said diverting code and said restoring code act by modifying said program counter value.

52. A computer program product as claimed in claim 43, comprising setting code operable to set a single step flag, said single step flag serving to limit hardware execution of variable length instructions to a single variable length instruction.

53. A computer program product as claimed in claim 52, wherein said single step flag serves to limit hardware execution of variable length instructions to a single variable length instruction from said concatenated instruction data to a single variable length instruction before control is returned to software to perform said step of restoring.

54. A computer program product as claimed in claim 52, wherein upon hardware execution of said single variable length instruction, said single step flag is cleared under hardware control.

55. A computer program product as claimed in claim 52, wherein said single step flag is stored within a coprocessor register.

56. A computer program product as claimed in claim 43, comprising calculating code operable to calculate a start address within said following region of memory of a following variable length instruction following said current variable length instruction.

57. A computer program product as claimed in claim 56, wherein said calculating code uses as inputs a start address of said following memory region and a program counter value pointing to said following variable length instruction within said fix-up memory region following execution of said current variable length instruction.

58. A computer program product as claimed in claim 57, comprising storing said start address of said following memory region before said diversion.

59. A computer program product as claimed in claim 43, wherein said variable length instructions are Javacard bytecode instructions executed as native instructions by said data processing apparatus.

60. A computer program product as claimed in claim 59, wherein said data processing apparatus also supports execution of instructions of a further instruction set, said steps of concatenating, diverting and restoring being performed under control of instructions of said further instruction set.

61. A computer program product as claimed in claim 60, wherein said diverting code and said restoring code use state switching branch instructions that serve to switch to execution of Java bytecodes starting from a specified memory address location.

62. A computer program product as claimed in claim 43, wherein a program to be executed is stored within fragmented memory regions within said memory.